Parallelism on Hybrid Metaheuristics for Vector Autoregression Models

Alfonso L. Castaño, Javier Cuenca, José-Matías Cutillas-Lozano, Domingo Giménez
Scientific Computing and Parallel Programming Group, University of Murcia

Jose J. López-Espín, Alberto Pérez-Bernabeu
Center of Operations Research, Miguel Hernández University

Outline

Context

- A project to apply to medical data techniques traditionally used in econometrics:
  Simultaneous Equation Models,
  Vector Autoregression Models (VAR),
  others...
- Big Data:
  large amounts of data, difficult to model by experts,
  design software tools to help experts.
- Work on computational aspects of VAR:
  Matrix computation,
  Metaheuristics,
  Parallel computing...

General ideas

- Dependent variables are analyzed by the model.
  The value of a variable at a time is linearly related to its past, but also to that of all other dependent variables in the model.
- Independent variables are used to predict endogenous variables.
  Their values at previous instants also appear in the model.

Model

$y^{(j)}$, $z^{(j)}$: vectors of dependent and independent variables at time j, $0 \leq j \leq t$.
$y^{(j)}$ depends on the value of y and z at t_y and t_z previous time points.
The linear dependence of $y^{(j)}$ with $y^{(j-r)}$ and $z^{(j-s)}$ is expressed by the matrices $A^{(r)}$ and $B^{(s)}$, $1 \leq r \leq t_y$, $1 \leq s \leq t_z$.
C denotes an independent vector.

$$y^{(j)} \approx y^{(j-1)}A^{(1)} + \ldots + y^{(j-t_y)}A^{(t_y)} + z^{(j-1)}B^{(1)} + \ldots + z^{(j-t_z)}B^{(t_z)} + C$$

$$\begin{pmatrix} y^{(t_y)} \\ \vdots \\ y^{(t)} \end{pmatrix} \approx \begin{pmatrix} y^{(t_y-1)} & \ldots & y^{(0)} & z^{(t_y-1)} & \ldots & z^{(t_y-t_z)} & 1 \\ & \vdots & & & \vdots & & \vdots \\ y^{(t-1)} & \ldots & y^{(t-t_y)} & z^{(t-1)} & \ldots & z^{(t-t_z)} & 1 \end{pmatrix} \begin{pmatrix} A_1 \\ \vdots \\ A_{t_y} \\ B_1 \\ \vdots \\ B_{t_z} \\ C \end{pmatrix}$$

Optimization problems arising

- Solve $\min_D ||Y - XD||_2$
  with Y the matrix of dependent variables,
  X the matrix of dependent and independent variables at earlier times,
  D the model to be obtained (A, B and C).
- Determine which variables are dependent and which independent.
- Determine the number of previous instants of time which gives the best prediction.
- Information criteria can be used to determine the preferred model: Akaike (AIC), Schwarz (BIC), Hannan-Quinn (HQC)...

Computational aspects

- Optimization:
  libraries (STATA),
  metaheuristics.
- Matrix computation:
  libraries (LAPACK),
  matrix decomposition (QR, LQ...),
  exploitation of the structure of the matrices (Toeplitz-like).
- Parallelism:
  in matrix computation,
  at the metaheuristic level,
  multilevel, hybrid, heterogeneous.

Hybridation of metaheuristics

Initialize(Sini,ParamIni)
Improve(Sini,Sref,ParamImpIni)
while not EndCondition(Sref,ParamEndCon) **do**
  Select(Sref,Ssel,ParamSel)
  Combine(Ssel,Scom,ParamCom)
  Diversify(Sref,Scom,Sdiv,ParamDiv)
  Improve(Scom,Sdiv,ParamImp)
  Include(Scom,Sdiv,Sref,ParamInc)
end while

General aspects of metaheuristics

- Each element of size $(d \cdot t_y + e \cdot t_z + 1) \cdot d$.
- Fitness computation: $O((t - t_y) \cdot (d \cdot t_y + e \cdot t_z + 1) \cdot d)$.
- The values of the Metaheuristic Parameters determine the metaheuristic, basic or hybrid.
- Basic metaheuristics:
  - Local Search: GRASP and Tabu Search.
  - Population based: Genetic and Scatter Search.

Local Search methods

- Neighborhood, $2 \cdot (d \cdot t_y + e \cdot t_z + 1) \cdot d$ elements, adding or subtracting to v a value in the interval $[0.001, v/10]$.
- If the best element in the neighborhood is better than the active element, it becomes the new active element.
- Intensification of the Improvement: maximum number of iterations.
- GRASP: after a number of iterations a new active element.
- Tabu List in improvement functions: the positions recently explored.
- Long-term Tabu Memory: select elements far from the mean of recent best elements.

Population Based methods

- Best and Worst elements are selected and combined to escape local minima.
- Combination by a simple Path Relinking: from ascendants v_1 and v_2, the two descendants are $\frac{1}{3}v_1 + \frac{2}{3}v_2$ and $\frac{2}{3}v_1 + \frac{1}{3}v_2$.
- Diversification (mutation) adding or subtracting to v a value in the interval $[0, v]$.
- Some nonbest elements are included for the next iteration.
- Depending on the parameters, the methods are closer or not to Genetic Algorithm or Scatter Search.

Metaheuristic Parameters

INEIni	Initial Number of Elements
PEIIni	Percentage of Elements to be Improved in the initialization
IIEIni	Intensification in the Improvement of Elements
LTLIni	Length of the Tabu List for local search
FNEIni	Final Number of Elements for successive iterations
MNIEnd	Maximum Number of Iterations
MIREnd	Maximum number of Iterations with Repetition
NBESel	Number of Best Elements selected for combination
NWESel	Number of Worst Elements selected for combination
NBBCom	Number of Best-Best elements combinations
NBWCom	Number of Best-Worst elements combinations
NWWCom	Number of Worst-Worst elements combinations
PEDDiv	Percentage of Elements to be Diversified
PEIImp	Percentage of Elements obtained by combination to be Improved
IIEImp	Intensification of the Improvement of Elements obtained by combination
IIDImp	Intensification of the Improvement of elements obtained by Diversification
LTLImp	Length of the Tabu List for local search on elements obtained by combination
LTDImp	Length of the Tabu list for local search on elements obtained by Diversification
NBEInc	Number of Best Elements included in the reference set for the next iteration
LTMInc	Long-Term Memory size for the selection of elements to be included in the reference set

Parallel metaheuristics

- Local Search methods: the amount of computation may not be enough for exploitation of parallelism;
- Population Based methods: the computation of the fitness can be costly;
- Multilevel parallelism:
  - Parallelization of the loops for treating the elements.
  - In the analysis of the neighborhood in the improvement functions.
  - Computation of the fitness function.

adaptation of the degree of parallelism at each level depending on the characteristics of the metaheuristic and the computational system.

Shared-Memory

- Independent parallelization of the basic functions.
- Parallelization of loops with OpenMP.
- Two-level parallelism in improvement functions: for the number of elements to improve, and for the neighborhood.
- Computation of fitness the main loops to be parallelized.

Multi GPU parallelism

- Island scheme: reference set divided in subsets assigned to different GPUs.
- Parallelization of the loops for the treatment of elements: steps assigned to different threads, and each thread calls to its GPU.
- GPUs working at the second level of the shared-memory version: the analysis of the neighborhood is done in parallel.
- The GPUs collaborate to compute each fitness.

CPU+GPU

InitializeCPU(Sini,ParamIni)
ComputefitnessGPU(Sini,ParamIni)
ImproveGPU(Sini,Sref,ParamImpIni) //two possible levels
while not EndConditionCPU(Sref,ParamEndCon) **do**
  SelectCPU(Sref,Ssel,ParamSel)
  CombineCPU(Ssel,Scom,ParamCom)
  ComputefitnessGPU(Scom,ParamCom)
  DiversifyCPU(Sref,Scom,Sdiv,ParamDiv)
  ComputefitnessGPU(Sdiv,ParamDiv)
  ImproveGPU(Scom,Sdiv,ParamImp) //two levels
  IncludeCPU(Scom,Sdiv,Sref,ParamInc)
end while

Computational nodes

- **marte**: 6 cores + 1 GPU
  AMD Phenom II X6 1075T + NVidia GeForce GTX 480 (Fermi)
- **saturno**: 24 cores + 1 GPU
  Intel Xeon E7530 + NVidia Tesla K20c (Kepler)
- **venus**: 12 cores + 1 GPU
  Intel Xeon E5-2620 + NVidia GeForce GT 640 (Kepler)
- **jupiter**: 12 cores + 6 GPUs
  Intel Xeon E5-2620 + 2 NVidia Tesla C2075 (Fermi) + 4 NVidia GeForce GTX 590 (Fermi)

Shared-Memory, Three levels of parallelism, Experiment

Analysis of parallelism at the three levels.

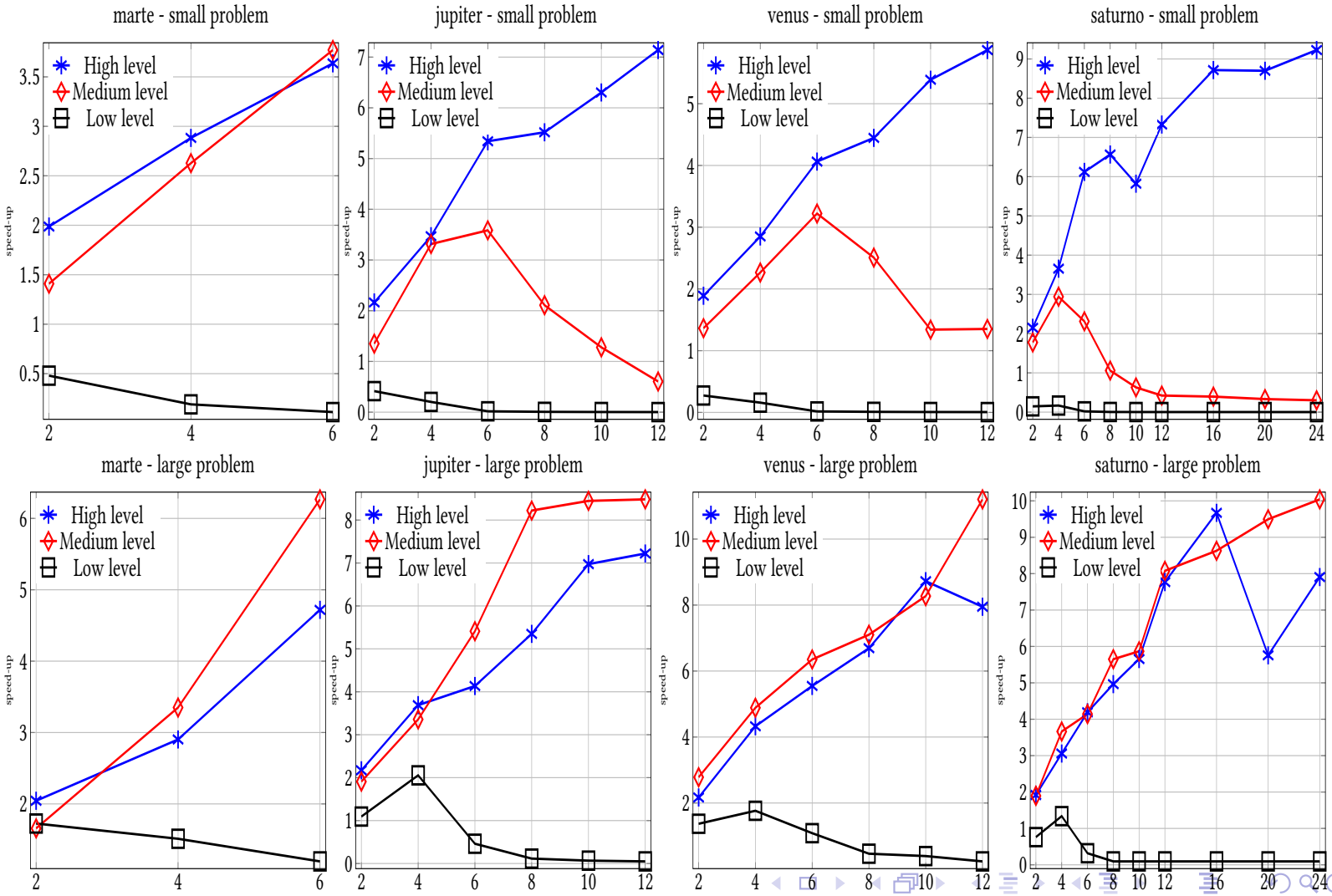
80	*INEIni*	Initial Number of Elements
50	*PEIIni*	Percentage of Elements to be Improved in the initialization
10	*IIEIni*	Intensification in the Improvement of Elements
1	*LTLIni*	Length of the Tabu List for local search
20	*FNEIni*	Final Number of Elements for successive iterations
10	*MNIEnd*	Maximum Number of Iterations
5	*MIREnd*	Maximum number of Iterations with Repetition
10	*NBESel*	Number of Best Elements selected for combination
5	*NWESel*	Number of Worst Elements selected for combination
10	*NBBCom*	Number of Best-Best elements combinations
10	*NBWCom*	Number of Best-Worst elements combinations
5	*NWWCom*	Number of Worst-Worst elements combinations
20	*PEDDiv*	Percentage of Elements to be Diversified
50	*PEIImp*	Percentage of Elements obtained by combination to be Improved
10	*IIEImp*	Intensification of the Improvement of Elements obtained by combination
20	*IIDImp*	Intensification of the Improvement of elements obtained by Diversification
1	*LTLImp*	Length of the Tabu List for local search on elements obtained by combination
1	*LTDImp*	Length of the Tabu list for local search on elements obtained by Diversification
15	*NBEInc*	Number of Best Elements included in the reference set for the next iteration
3	*LTMInc*	Long-Term Memory size for the selection of elements to be included in the reference set

Small Problem: $t = 200$, $d = 4$, $e = 2$, $t_y = 3$, $t_z = 2$.

Large Problem: $t = 600$, $d = 8$, $e = 6$, $t_y = 7$, $t_z = 6$.

Shared-Memory, Three levels of parallelism, Speed-up

Low level parallelism is far worse than the others

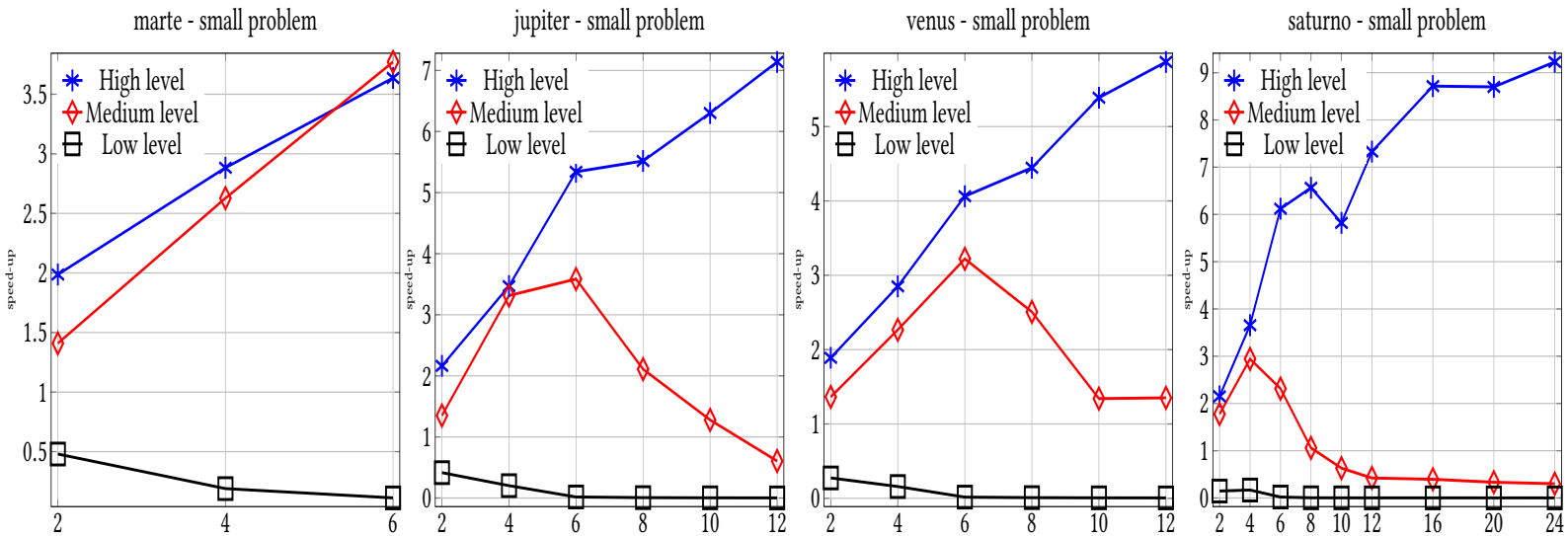


Shared-Memory, Three levels of parallelism, Speed-up

SP:

preferable exploitation of parallelism at the highest level

performance with medium level parallelism decreases when the number of threads increases

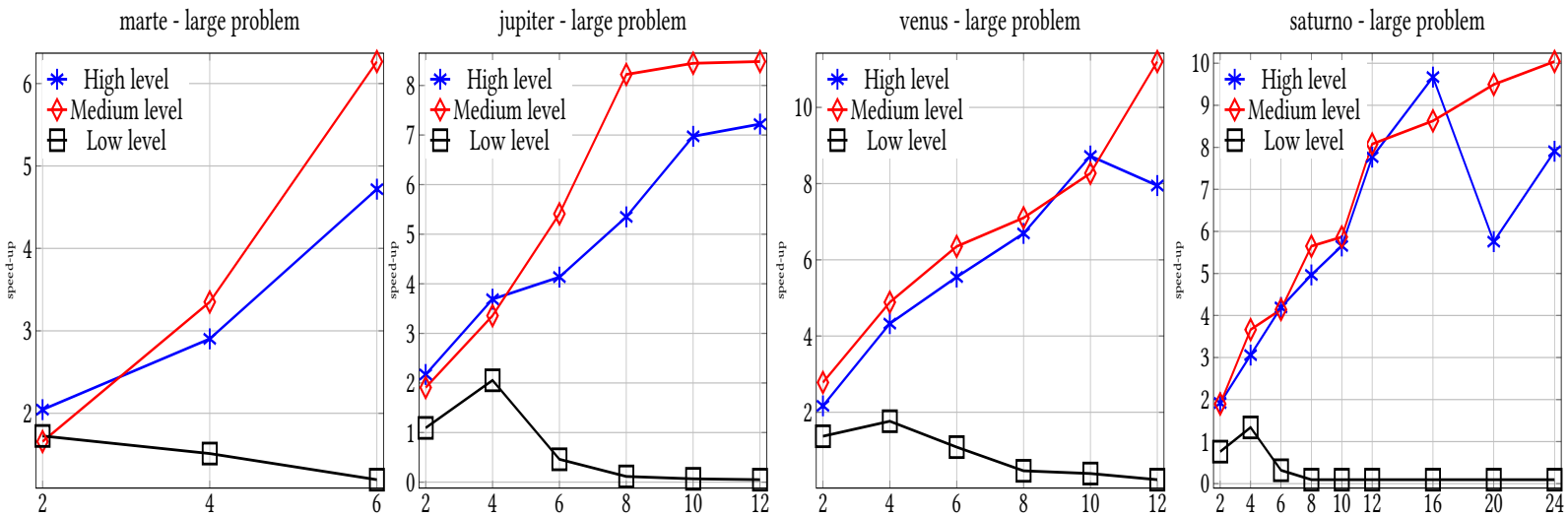


Shared-Memory, Three levels of parallelism, Speed-up

LP:

the speed-up with medium and large parallelism is similar

medium-level slightly better



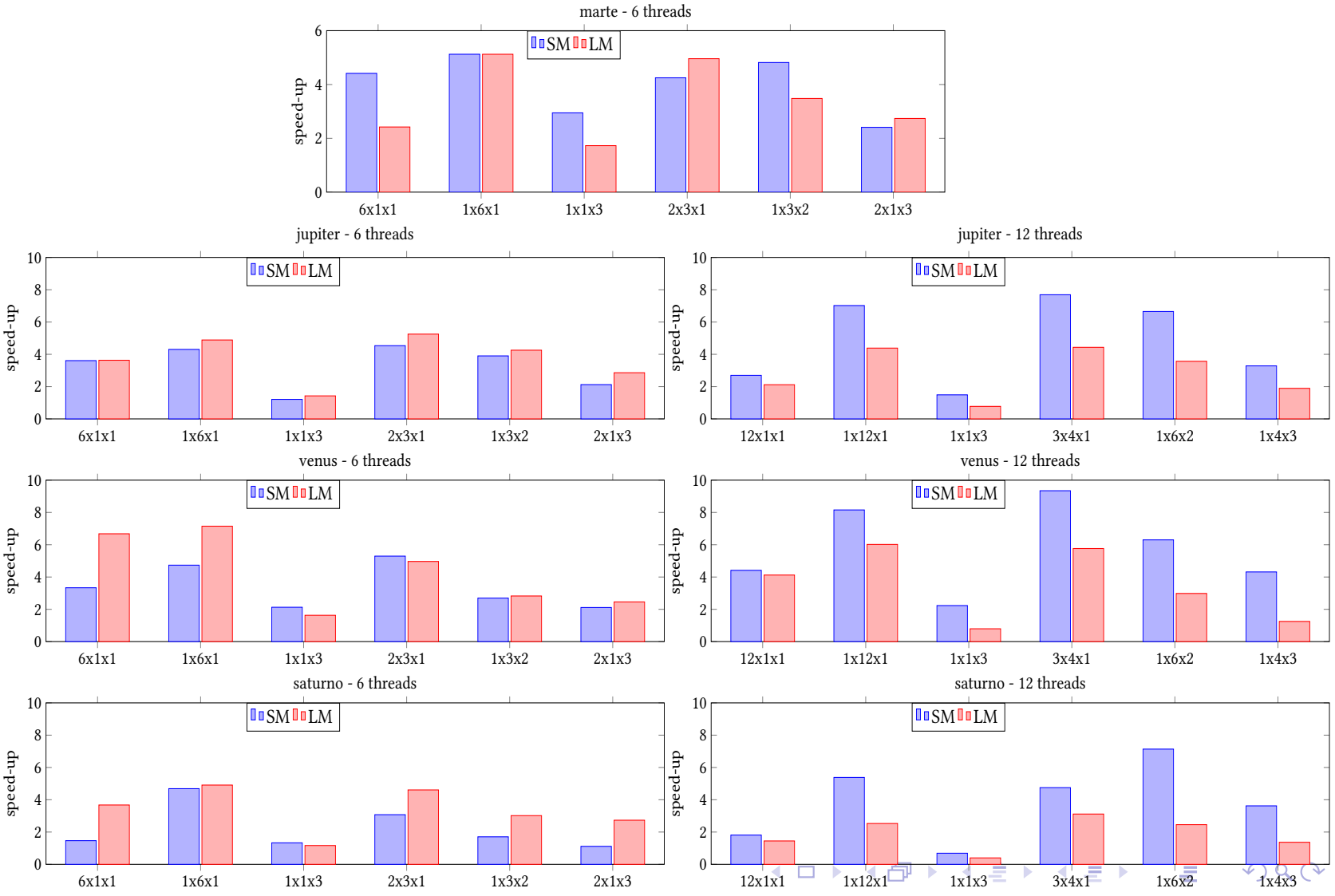
Shared-Memory, Combination of parallelism levels

Experiments: large time series;
small metaheuristic $(10, 5, 50, 5, 1, 2, 1, 2, 1, 0, 50, 5, 1, 10, 5, 1, 3, 1)$
large metaheuristic $(20, 10, 50, 5, 1, 4, 2, 4, 2, 0, 50, 5, 1, 10, 5, 1, 6, 1)$

- Small metaheuristic propitiates the use of parallelism at a low level.
- Large metaheuristic: better performance with 6 threads than with 12. More memory accessed by more threads?
- The speed-up with 12 threads is far from the optimum.

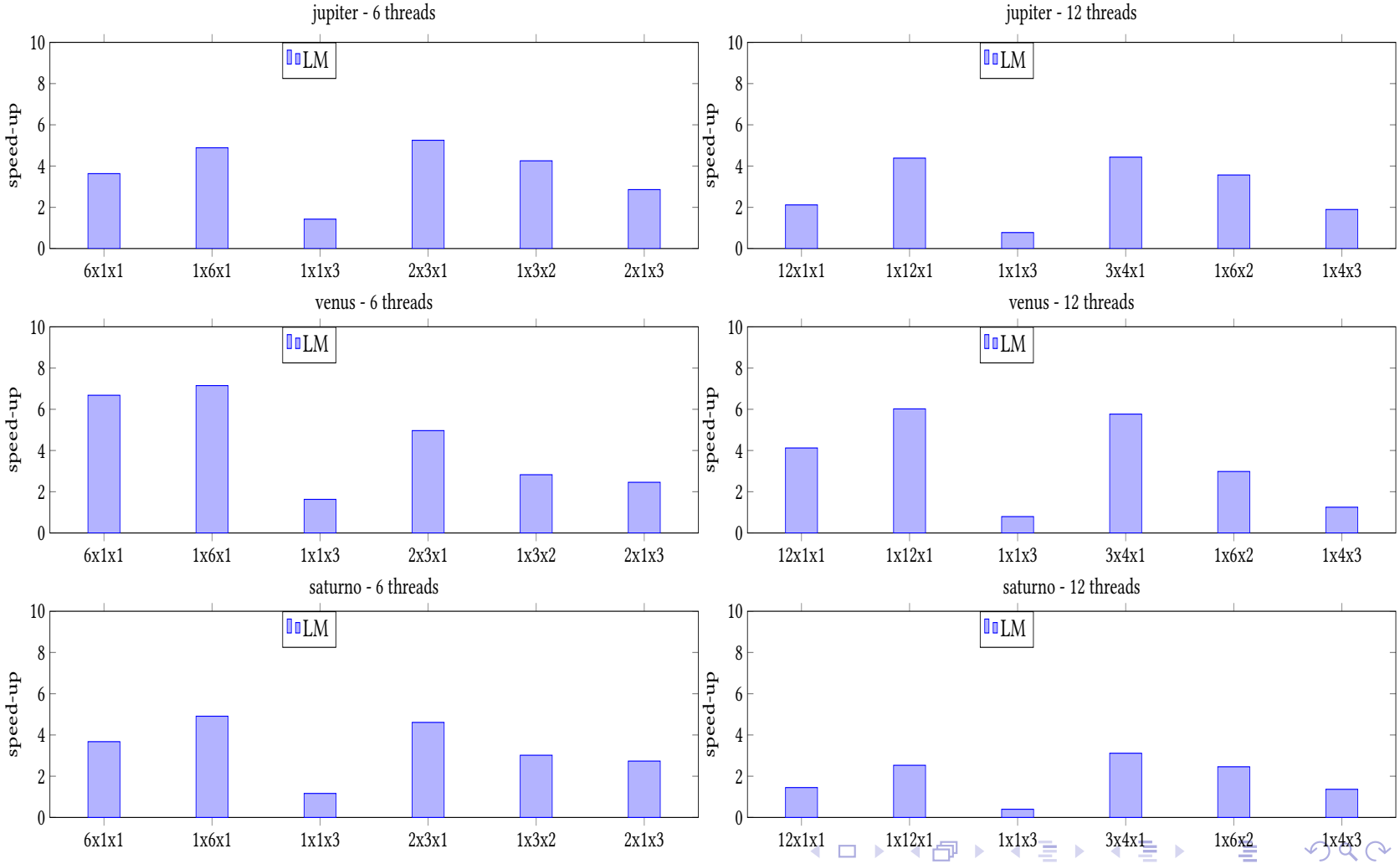
Shared-Memory, Combination of parallelism levels

The best option is not parallelism at the highest level



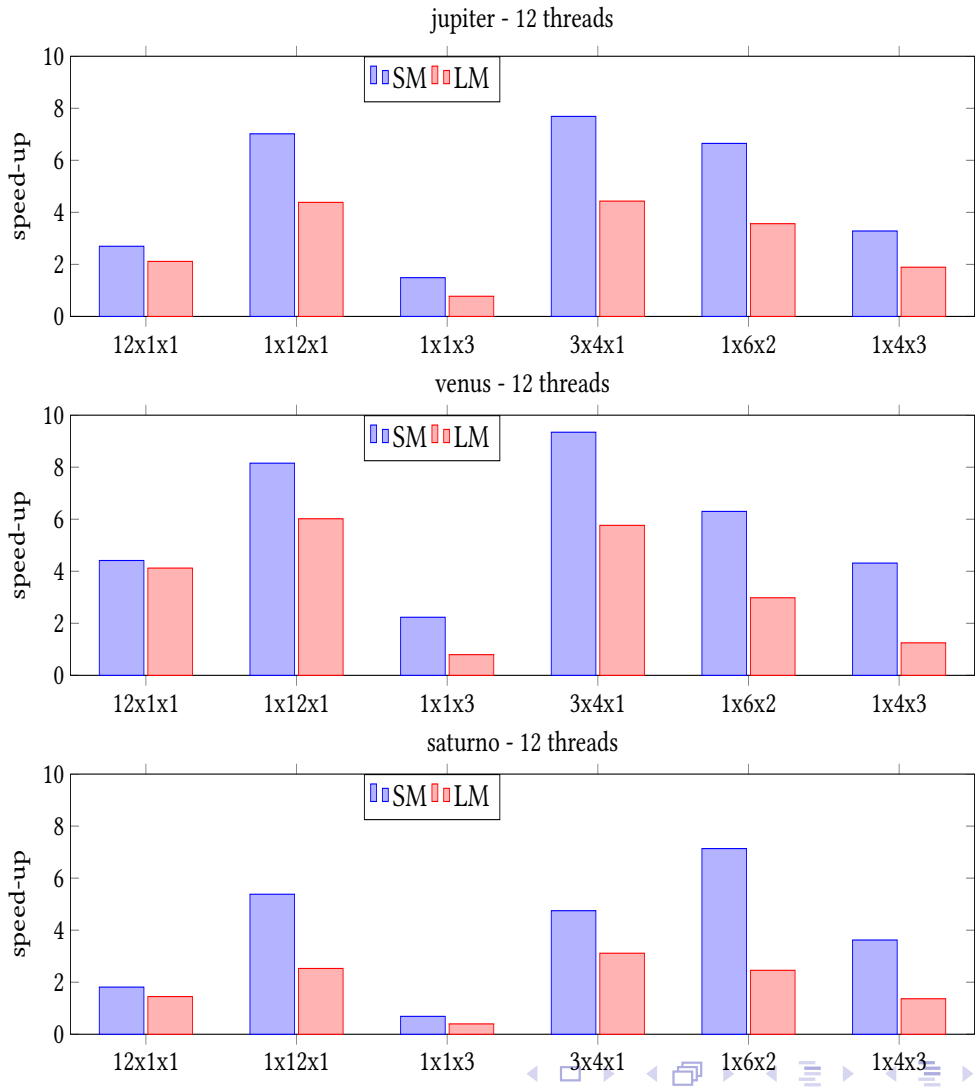
Shared-Memory, Combination of parallelism levels

Large metaheuristic: better performance with 6 threads than with 12. More memory accessed by more threads?



Shared-Memory, Combination of parallelism levels

The speed-up with 12 threads is far from the optimum



GPU, preliminary results

In **jupiter**

- witn 1 GPU, computation of the fitness in GPU, $t = 200$:

	CPU/GPU
$d = 4,\ e = 2,\ t_y = 3,\ t_z = 2$	0.83
$d = 5,\ e = 3,\ t_y = 3,\ t_z = 2$	0.92
$d = 5,\ e = 3,\ t_y = 4,\ t_z = 2$	1.25

- with 4 GPU (2 Tesla + 2 GeForce), with $t = 500$:

	time
homogeneous distribution	1854
heterogeneous distribution	2249
dynamic distribution	1598

Conclusions

- Matrix representation of VAR.
- Application of hybrid metaheuristics.
- Parallelism:
  - Shared-memory version, blocked when the number of threads increases.
  - Hybrid CPU+GPU version, beneficial only for large problems.

Future works

- Optimization of GPU versions.
- Application of the hybrid parallel schema to other problems (molecule docking).
- Methodology to decide the functions and level at which the GPUs are used.
- Detailed analysis of influence of parallelism on the fitness.
- Combination of matrix computation (LAPACK) and metaheuristics.
- Exploit Toeplitz-like structure of the matrices.
- Restrictions in the model: interval of integers, interval of floats, set of floats...
- Application to real problems in different fields (medical and economic data).
- The work is part of a project applying econometric techniques to model medical data, so the models obtained with VARs will be compared with those with other techniques (SEM) and software (STATA).

Parallelism on Hybrid Metaheuristics for Vector Autoregression Models

Questions?

Improving the fitness

- Metaheuristics:
  M1: reference set of 100 elements, 150 combinations, 20% mutation
  M2: reference set of 20 elements, 290 combinations, no mutations
- **jupiter**, 12 cores